

10070 = 10103

Date of Deposit:

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date inscribed above and is addressed to the Assistant Commissioner of Patents, Box PATENT APPLICATION, Washington, D.C. 20231.

(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

-1-

Field of the Invention

5 The present invention relates generally to the storage and data networking fields, and more particularly, relates to a scheduler, scheduling method, and computer program product for implementing Quality-of-Service (QoS) scheduling with a cached status array.

Related Applications

Related United States patent applications by William John
10 Goetzinger, Glen Howard Handlogten, James Francis Mikos, and David Alan
Norgaard and assigned to the present assignee are being filed on the same
day as the present patent application including:

United States patent application Serial Number _____,
entitled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING PEAK
15 SERVICE DISTANCE USING NEXT PEAK SERVICE TIME VIOLATED
INDICATION":

United States patent application Serial Number _____
entitled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING
QUALITY OF SERVICE WITH AGING TIME STAMPS";

20 United States patent application Serial Number _____
entitled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING

ROC920010205US1

QUALITY OF SERVICE ANTICIPATING THE END OF A CHAIN OF FLOWS";

United States patent application Serial Number _____,
entitled "WEIGHTED FAIR QUEUE HAVING EXTENDED EFFECTIVE
5 RANGE";

United States patent application Serial Number _____,
entitled "WEIGHTED FAIR QUEUE SERVING PLURAL OUTPUT PORTS";

United States patent application Serial Number _____,
entitled "WEIGHTED FAIR QUEUE HAVING ADJUSTABLE SCALING
10 FACTOR"; and

United States patent application Serial Number _____,
entitled "EMPTY INDICATORS FOR WEIGHTED FAIR QUEUES".

Description of the Related Art

Storage and data networks are designed to support the integration of
15 high quality voice, video, and high speed data traffic. Storage and data
networking promises to provide transparent data sharing services at high
speeds. It is easy to see that rapid movement and sharing of diagrams,
pictures, movies, audio, and the like requires tremendous bandwidth.
Network management is concerned with the efficient management of every
20 bit of available bandwidth.

A need exists for a high speed scheduler for networking that ensures
the available bandwidth will not be wasted and that the available bandwidth
will be efficiently and fairly allocated. The scheduler should permit many
network traffic flows to be individually scheduled per their respective
25 negotiated Quality-of-Service (QoS) levels. This would give system
administrators the ability to efficiently tailor their gateways, switches, storage
area networks (SANs), and the like. Various QoS can be set up using
combinations of precise guaranteed bandwidth, required by video for
example, and limited or unlimited best effort bandwidth for still pictures,
30 diagrams, and the like. Selecting a small amount of guaranteed bandwidth

ROC920010205US1

1000440-1.00001

with the addition of some bandwidth from the pool of best effort bandwidth should guarantee that even during the highest peak periods, critical data will be delivered to its application at that guaranteed rate.

- 5 A scheduler advantageously may be added to a network processor to enhance the quality of service (QoS) provided by the network processor subsystem.

- 10 Known high-performance network processor scheduler systems are able to search entire calendars in one system cycle for the purpose of updating calendar status, that is active flow status. High-performance schedulers will no longer be able to search entire calendar arrays within one system cycle as performance requirements increase. Bandwidth constraints no longer allow the entire calendar array to be searched each cycle. A new technique is needed to perform calendar updates.

Summary of the Invention

- 15 A principal object of the present invention is to provide a QoS scheduler, scheduling method, and computer program product for implementing Quality-of-Service (QoS) scheduling with a cached status array. Other important objects of the present invention are to provide such QoS scheduler, scheduling method, and computer program product for
- 20 implementing Quality-of-Service (QoS) scheduling with a cached status array substantially without negative effect and that overcome some disadvantages of prior art arrangements.

- 25 In brief, a QoS scheduler, scheduling method, and computer program product are provided for implementing Quality-of-Service (QoS) scheduling with a cached status array. A plurality of calendars are provided for scheduling the flows. An active flow indicator is stored for each calendar entry in a calendar status array (CSA). A subset of the active flow indicators from the calendar status array (CSA) is stored in a cache. The calendar status array (CSA) is updated based upon a predefined calendar range and
- 30 resolution. The subset of the active flow indicators from the calendar status array (CSA) is used to determine a given calendar for servicing.

In accordance with features of the invention, the cache copy subset of the active flow indicators from the calendar status array (CSA) is used to increment a current pointer (CP) by an identified number of positions up to a current time (CT) value, where the identified number of positions is equal to a variable number of inactive flow indicators up to the current time (CT) value and the identified number of positions has a maximum value equal to a number of entries in the cache.

Brief Description of the Drawings

The present invention together with the above and other objects and advantages may best be understood from the following detailed description of the preferred embodiments of the invention illustrated in the drawings, wherein:

FIG. 1A is a block diagram illustrating a network processor system including a scheduler for carrying out scheduling methods for implementing Quality-of-Service (QoS) scheduling with a cached status array of the preferred embodiment;

FIGS. 1B is diagram providing a graphical illustration of various types of QoS algorithms in accordance with the preferred embodiment;

FIG. 2 is a high-level system diagram illustrating the scheduler for carrying out scheduling methods for implementing QoS scheduling with a cached status array of the preferred embodiment;

FIG. 3 is a diagram illustrating a calendar status array of the preferred embodiment;

FIGS. 4 and 5 are diagrams illustrating a calendar array including multiple epochs of the preferred embodiment; and

FIG. 6 is a block diagram illustrating a computer program product in accordance with the preferred embodiment.

Detailed Description of the Preferred Embodiments

ROC920010205US1

Having reference now to the drawings, in FIG. 1A, there is shown a network processor system generally designated by the reference character 100 including a scheduler 200 for carrying out scheduling methods for implementing Quality-of-Service (QoS) scheduling with a cached status array of the preferred embodiment. As shown in FIG. 1A, network processor system 100 includes a network processor 102 that executes software responsible for forwarding network traffic. Network processor 102 includes hardware assist functions for performing operations, such as table searches, policing, and statistics tracking. A dataflow 104 serves as the primary data path for transmitting and receiving data flow traffic, for example, via a network interconnect 106 and/or a switch fabric interface 108. Dataflow 104 provides an interface to a large data store memory 110 for buffering of traffic bursts when an incoming frame rate exceeds an outgoing frame rate. An external flow queue memory 112 is coupled to scheduler 200. As performance of network processor 102 continues to increase, unique techniques and design solutions enable the QoS scheduler 200 to perform reliably at these high data rates.

Scheduler 200 of the preferred embodiment permits many network traffic flows, for example, 64 thousand (64K) network traffic flows to be individually scheduled per their respective assigned Quality-of-Service (QoS) level. Each flow is basically a one-way connection between two different points. QoS parameters are held in a flow queue control block (FQCB), such as in the external flow queue memory 112. QoS parameters include sustained service distance (SSD), peak service distance (PSD), queue distance (QD), port identification (ID), and the like. There can be, for example, 64 thousand flows and a FQCB for each flow.

FIG. 1B provides a graphical illustration of various types of QoS algorithms. The scheduler 200 provides for quality of service by maintaining flow queues that may be scheduled using various algorithms, such as a set guaranteed bandwidth, or best effort or weighted fair queue (WFQ) with or without a peak bandwidth service (PBS) limit. The best effort or weighted fair queue is limited via the peak service distance (PSD) QoS parameter. The guaranteed bandwidth is set via the sustained service distance (SSD) QoS parameter. A combination of these algorithms provide efficient utilization of available bandwidth. The scheduler 200 supplements the

congestion control algorithms of dataflow 104 by permitting frames to be discarded based on per flow queue thresholds.

Referring now to FIG. 2, there is shown a high-level system diagram illustrating the scheduler 200 for carrying out scheduling methods of the preferred embodiment. Scheduler 200 includes a bus interface 202 coupled to a system bus 204 interconnecting modules in the system 100. Chipset messages are exchanged between modules using system bus 204. Messages include flow enqueue requests which add frames to a given flow and read and write requests. Scheduler 200 includes a message buffer 206, such as a first-in first-out (FIFO) message buffer, that stores messages until they are ready to be executed. Scheduler 200 includes a queue manager 208 coupled to the message buffer 206. Queue manager 208 processes the incoming messages to determine what action is required. Queue manager 208 is coupled to calendars and rings block 220 and a memory manager 224. A winner partition 222 arbitrates between the calendars and rings 220 to choose which flow will be serviced next. The memory manager 224 coordinates data reads from and writes to a first and second external static random access memory (SRAM) 226 and 228 and an internal memory array 230.

For a flow enqueue request received by queue manager 208, the flow's FQCB information is retrieved from one of the external SRAM 226 or 228 or internal array 230 and examined to determine if the new frame should be added to an existing frame string for a given flow, start a new frame string, or be discarded. In addition, the flow queue may be attached to a calendar or ring for servicing in the future. Read and write request messages received by queue manager 208 are used to initialize flows.

Port back-pressure from the dataflow 104 to the scheduler 200 occurs via the port status request message originated from the dataflow and applied to the calendar and rings block 220. When a port threshold is exceeded, all WFQ and PBS traffic associated with that port is held in the scheduler 200 and the selection logic of winner partition 222 does not consider those flows potential winners. When port back-pressure is removed, the flows associated with that port are again eligible to be winners.

5

10

25

30

containing FQCB aging information is used with logic that searches through the flows and invalidates expired time stamps.

Queue manager 208 performs the queuing operation of scheduler 200 generally as follows: A linked list or string of frames is associated with each flow. Frames are always enqueued to the tail of the linked list. Frames are always dequeued from the head of the linked list. Flows are attached to one or more of four calendars/rings (LLS, NLS, PBS, WFQ) 220 using the QoS parameters. Selection of which flow to service is done by examining the calendars/rings 220 in the order of LLS, NLS, PBS, WFQ. Then the frame at the head of the selected flow is selected for service. The flow queues are not grouped in any predetermined way to target port. The port number for each flow is user programmable. All WFQ flows with the same port ID are attached to the same WFQ ring. The QoS parameters also apply to the discard flow. The discard flow address is user selectable and is set up at configuration time.

When a flow enqueue request is sent to the scheduler 200, its frame is tested for possible discard using information from the flow enqueue request message and information stored in the FQCB. If the frame is to be discarded then the FQCB pointer is changed from the FQCB in flow enqueue request message to the discard FQCB. Alternatively, the frame is added to the tail end of the FCB chain associated with the FQCB. In addition, the flow is attached if it is not already attached to the appropriate calendar (LLS, NLS, PBS), or ring (WFQ). As time passes, selection logic of winner partition 222 determines which flow is to be serviced (first LLS, then NLS, then PBS, then WFQ). If a port bandwidth threshold has been exceeded, the WFQ and PBS component associated with that port are not eligible to be selected. When a flow is selected as the winner, the frame at the head of the FCB chain for the flow is dequeued and a port enqueue response message is issued to the dataflow 104. If the flow is eligible for a calendar reattach, the flow is reattached to the appropriate calendar (LLS, NLS, PBS) or ring (WFQ) in a manner consistent with the QoS parameters.

Scheduler 200 of the preferred embodiment keeps track of multiple calendars 220. For example, calendars 220 include 5 epochs of low latency service (LLS) calendars, 5 epochs of normal latency service (NLS)

5

10

25

30

10

15

25

30

entries (120:127) from the previous cache copy access of CSA 300 that are shifted to a low portion of cache 302 and 24 active flow indicator bits corresponding to calendar entries (128:151) in a top portion of cache 302 from one read of CSA 300.

5 The cached copy data in cache 302 is also used to allow the current pointer (CP) to catch-up to current time (CT). The CP can regularly fall behind CT. For example, when servicing a calendar entry that contains several chained flows, the CT increments once each time a flow is serviced. The CP does not increment until all flows are serviced in a chain 402 of
10 multiple flows for FLOW ID as shown in FIG. 4, where the CP is stored in the FLOW ID flow queue control block (FQCB). Without the ability to increment CP at a rate faster than CT is incremented, CP will fall extremely far behind CT causing system problems.

15 In accordance with features of the preferred embodiment, CP is allowed to be incremented more than 1 position per tick by utilizing the 32 bit entry cache 302. CP is allowed to be incremented by an identified number of calendar entries having no active flows attached up to the CT. To illustrate this, an Example 1 is shown where examining the 32 bit cache 302, a window within the cache reveals the relationship between CP and CT as
20 follows:

Example 1 of Cache window

entry 317	1	
entry 316	0	← CT
entry 315	0	
25 entry 314	0	
entry 313	0	
entry 312	1	← CP

30 In Example 1, CP is currently pointing to a calendar entry with an active flow, for example, entry 312 as shown. Once this flow is serviced, CP may be incremented by 4 since the indicator bits between CT and CP+4 are 0. This indicates that no active flows are attached to the next 4 calendar entries. By incrementing CP by 4, for example, to calendar entry 316, CP will once again = CT. CP is never incremented past CT. This technique

allows CP to be incremented by at most 32 positions each tick, that is, the number of entries in the cache 302.

In accordance with features of the preferred embodiment, calendars 220 are segmented into epochs. Epoch is a term used to identify a technique that increases the effective range of a calendar 220 without increasing the physical size of the on-chip array CSA 300 by segmenting the calendar into sections or epochs. Epoch 0 has the highest resolution and lowest range. Epochs 1 through p will have a range of $n(p)$ times the range of the first epoch and a resolution of $1/n^{**p}$ of the first epoch, where n equals a set scaling factor. As the epoch number increases, calendar range is extended and resolution is reduced.

Referring now to FIG. 4, the low latency calendar 220 is shown including multiple epochs generally designated by reference numeral 400. Epoch 0, 400 has the highest resolution and the lowest range. Epoch 1 through Epoch 4 have a range from 16 to 16^{**4} or 65,536 times the range of the first Epoch 0. The resolution of Epoch 1 through Epoch 4 is represented by $1/16^{\text{Epoch}}$; that is Epoch 1 has a resolution of $1/16$ of Epoch 0 and Epoch 4 has a resolution of $1/16^4$ of Epoch 0.

FIG. 5A illustrates the preferred embodiment of calendar epochs generally designated by reference numeral 500 including epochs 0-p, where $p=4$. Based on this embodiment, since the lower calendar epochs will go through their cached data more quickly, the lower calendar epochs access the CSA 300 more frequently than the higher epochs, for example as follows:

Epoch 0s are accessed every 4 ticks
Epoch 1s are accessed every 16 ticks
Epoch 2s are accessed every 64 ticks
Epoch 3s are accessed every 256 ticks
Epoch 4s are accessed every 1024 ticks

Referring now to FIG. 5B, the number of ticks between epoch accesses is determined based upon the scaling factor of the epoch and number of calendars used. The example above is for a case of 3 calendars

(LLS, NLS and PS). Each of the calendars has 5 epochs with a scaling factor of 16, as illustrated in FIG. 5A. An exemplary algorithm 510 for accessing the CSA 300 for updating the cache copy data of cache 302 is:

CSA_Calendar_Update <=

```

5      LLSEpoch0 when CSA_Access_Counter (1 Downto 0) = 00      Else
      NLSEpoch0 when CSA_Access_Counter (1 Downto 0) = 01      Else
      PSEpoch0  when CSA_Access_Counter (1 Downto 0) = 10      Else
10     LLSEpoch1 when CSA_Access_Counter (3 Downto 0) = 0011    Else
      NLSEpoch1 when CSA_Access_Counter (3 Downto 0) = 0111    Else
      PSEpoch1  when CSA_Access_Counter (3 Downto 0) = 1011    Else
      LLSEpoch2 when CSA_Access_Counter (5 Downto 0) = 001111   Else
      NLSEpoch2 when CSA_Access_Counter (5 Downto 0) = 011111   Else
15     PSEpoch2  when CSA_Access_Counter (5 Downto 0) = 101111   Else
      LLSEpoch3 when CSA_Access_Counter (7 Downto 0) = 00111111  Else
      NLSEpoch3 when CSA_Access_Counter (7 Downto 0) = 01111111  Else
      PSEpoch3  when CSA_Access_Counter (7 Downto 0) = 10111111  Else
      LLSEpoch4 when CSA_Access_Counter (9 Downto 0) = 0011111111 Else
20     NLSEpoch4 when CSA_Access_Counter (9 Downto 0) = 0111111111 Else
      PSEpoch4  when CSA_Access_Counter (9 Downto 0) = 1011111111 Else
      None

```

In addition to the regularly scheduled read of the CSA 300 for a given calendar cache 302, the cache 302 can be updated by snooping the CSA location for an enqueue or reattach event. This gets the data to the cache 302 in the event that the normally scheduled update would not get the data to the cache in time. Also if the same flow is picked twice in a row as a winner from the same calendar or ring, the CSA 300 will not be updated as there is not enough time to add and remove the bit from the CSA. Special hardware is used in this case to properly schedule the flow.

Referring now to FIG. 6, an article of manufacture or a computer program product 600 of the invention is illustrated. The computer program product 600 includes a recording medium 602, such as, a floppy disk, a high capacity read only memory in the form of an optically read compact disk or CD-ROM, a tape, a transmission type media such as a digital or analog communications link, or a similar computer program product. Recording medium 602 stores program means 604, 606, 608, 610 on the medium 602 for carrying out scheduling methods for implementing Quality-of-Service

5

10

While the present invention has been described with reference to the details of the embodiments of the invention shown in the drawing, these details are not intended to limit the scope of the invention as claimed in the appended claims.